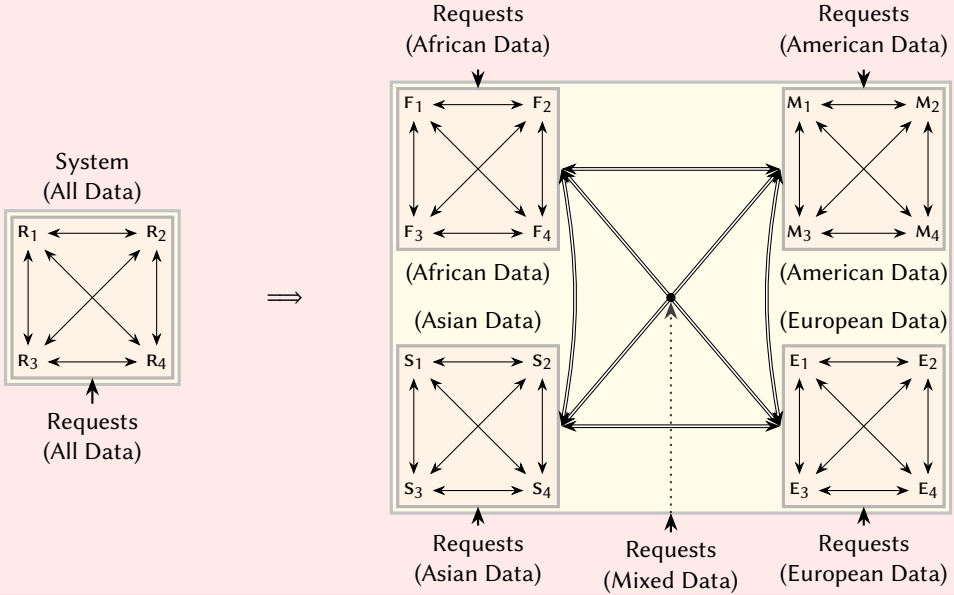
Efficient Fault-Tolerant Cluster-Sending: *Reliable and Efficient Communication between Byzantine Fault-Tolerant Clusters*

Jelle Hellings[1] Mohammad Sadoghi[2]

[1]Department of Computing and Software
McMaster University

[2]Exploratory Systems Lab
Department of Computer Science
University of California, Davis

Motivation: High-performance resilient system



Blockchain interoperability in the wild

Permissionless blockchains

- Bitcoin, Ethereum, ...
- Proof-of-Work, Proof-of-Stake, ...
- BTC Relay.
- Cosmos, Polkadot.
- Atomic swaps, atomic commitment, cross-chain deals, ...

Blockchain interoperability in the wild

Permissionless blockchains

- Bitcoin, Ethereum, ...
- Proof-of-Work, Proof-of-Stake, ...
- BTC Relay.
- Cosmos, Polkadot.
- Atomic swaps, atomic commitment, cross-chain deals, ...

Permissioned blockchains

- ResilientDB, BFT-SMaRt, ...
- Practical Byzantine Fault Tolerance.
- Steward, GeoBFT.
- AHL, ByShard, Chainspace, Cerberus, RingBFT.
- *The cluster-sending problem.*

The Byzantine cluster-sending problem

The problem of sending a value v from a cluster C_1 to a cluster C_2 such that

- all non-faulty replicas in C_2 RECEIVE the value v;
- all non-faulty replicas in C_1 CONFIRM that the value v was received; and
- C_2 only receives a value v if all non-faulty replicas in C_1 AGREE upon sending v.

Requirements to provide reliable communication between clusters with Byzantine replicas.

Global communication versus local communication

Straightforward cluster-sending solution (crash failures)

Pair-wise broadcasting with $(\mathbf{f}_1 + 1)(\mathbf{f}_2 + 1) \approx \mathbf{f}_1 \times \mathbf{f}_2$ messages.

	*Ping round-trip times (*ms*)*						*Bandwidth (*Mbit/s*)*					
	OR	IA	Mont.	BE	TW	Syd.	OR	IA	Mont.	BE	TW	Syd.
Oregon	≤ 1	38	65	136	118	161	7998	669	371	194	188	136
Iowa		≤ 1	33	98	153	172		10004	752	243	144	120
Montreal			≤ 1	82	186	202			7977	283	111	102
Belgium				≤ 1	252	270				9728	79	66
Taiwan					≤ 1	137					7998	160
Sydney						≤ 1						7977

Lower bounds for cluster-sending: Example

$\mathbf{n}_1 = 15$ $\mathbf{f}_1 = 7$

$\mathbf{n}_2 = 5$ $\mathbf{f}_2 = 2$

Claim (crash failures)

Any correct algorithm needs to send at least 14 messages.

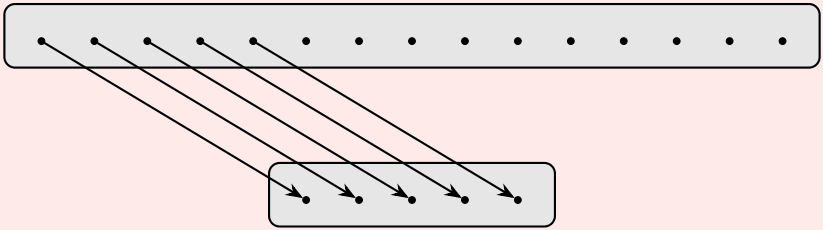
Lower bounds for cluster-sending: Example

$\mathbf{n}_1 = 15$ $\mathbf{f}_1 = 7$

$\mathbf{n}_2 = 5$ $\mathbf{f}_2 = 2$

Claim (crash failures)

Any correct algorithm needs to send at least 14 messages.



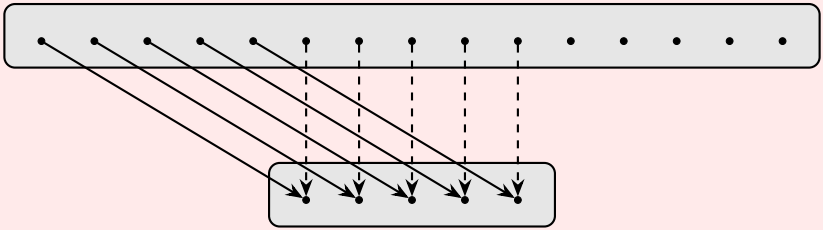
Lower bounds for cluster-sending: Example

$\mathbf{n}_1 = 15$ $\mathbf{f}_1 = 7$

$\mathbf{n}_2 = 5$ $\mathbf{f}_2 = 2$

Claim (crash failures)

Any correct algorithm needs to send at least 14 messages.



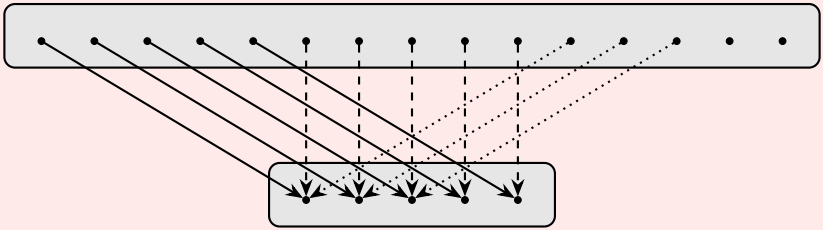
Lower bounds for cluster-sending: Example

$\mathbf{n}_1 = 15$ $\mathbf{f}_1 = 7$

$\mathbf{n}_2 = 5$ $\mathbf{f}_2 = 2$

Claim (crash failures)

Any correct algorithm needs to send at least 14 messages.



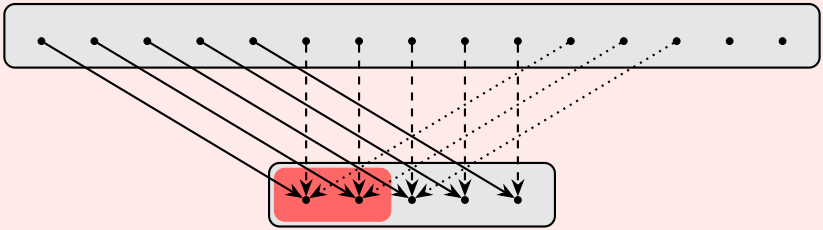
Lower bounds for cluster-sending: Example

$\mathbf{n}_1 = 15$ $\mathbf{f}_1 = 7$

$\mathbf{n}_2 = 5$ $\mathbf{f}_2 = 2$

Claim (crash failures)

Any correct algorithm needs to send at least 14 messages.



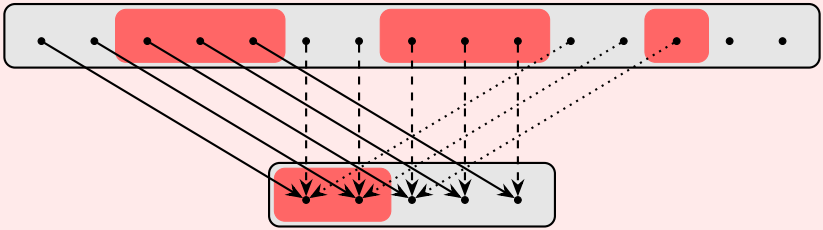
Lower bounds for cluster-sending: Example

$\mathbf{n}_1 = 15$ $\mathbf{f}_1 = 7$

$\mathbf{n}_2 = 5$ $\mathbf{f}_2 = 2$

Claim (crash failures)

Any correct algorithm needs to send at least 14 messages.



Lower bounds for cluster-sending: Results

Theorem (Cluster-sending lower bound, simplified)

We need to exchange $\max(\mathbf{n}_1, \mathbf{n}_2)$ *messages to do cluster-sending.*

Theorem (Cluster-sending lower bound, crash failures)

Assume $\mathbf{n}_1 \geq \mathbf{n}_2$ *and let*

$$q = (\mathbf{f}_1 + 1) \operatorname{div} \mathbf{nf}_2; \qquad r = (\mathbf{f}_1 + 1) \bmod \mathbf{nf}_2.$$

We need to exchange at least $q\mathbf{n}_2 + r + \mathbf{f}_2 \operatorname{sgn} r \approx \mathbf{n}_1$ *messages to do cluster-sending.*

Cluster-sending via bijective sending

Protocol for the sending cluster C_1:

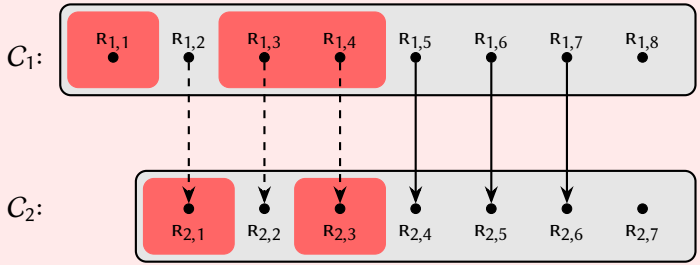
1: All replicas in $\mathrm{nf}(C_1)$ agree on v and construct $\langle v \rangle_{C_1}$.
2: Choose replicas $S_1 \subseteq C_1$ with $\mathbf{n}_{S_1} = \mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1$.
3: Choose replicas $S_2 \subseteq C_2$ with $\mathbf{n}_{S_2} = \mathbf{f}_{C_1} + \mathbf{f}_{C_2} + 1$.
4: Choose a bijection $b : S_1 \to S_2$.
5: **for** $\mathrm{R}_1 \in S_1$ **do**
6: $\quad \mathrm{R}_1$ sends $(v, \langle v \rangle_{C_1})$ to $b(\mathrm{R}_1)$.

Protocol for the receiving cluster C_2:

7: **event** $\mathrm{R}_2 \in \mathrm{nf}(C_2)$ receives $(w, \langle w \rangle_{C_1})$ from $\mathrm{R}_1 \in C_1$ **do**
8: $\quad$ Broadcast $(w, \langle w \rangle_{C_1})$ to all replicas in C_2.
9: **event** $\mathrm{R}'_2 \in \mathrm{nf}(C_2)$ receives $(w, \langle w \rangle_{C_1})$ from $\mathrm{R}_2 \in C_2$ **do**
10: $\quad \mathrm{R}'_2$ considers w *received*.

Cluster-sending via bijective sending—visualized

$\mathbf{n}_1 = 8, \mathbf{n}_2 = 7, \mathbf{f}_1 = 3, \mathbf{f}_2 = 2, \sigma = 6$



Cluster-sending: Can we do better?

Pessimistic

No: these algorithms are worst-case optimal.
Cannot do better than *linear communication* in the size of the clusters.

Cluster-sending: Can we do better?

Pessimistic

No: these algorithms are worst-case optimal.
Cannot do better than *linear communication* in the size of the clusters.

Probabilistic approaches

Yes: if we randomly choose sender and receiver, then we often do much better!

Cluster-sending via probabilistic cluster-sending

Protocol for the sending cluster C_1:

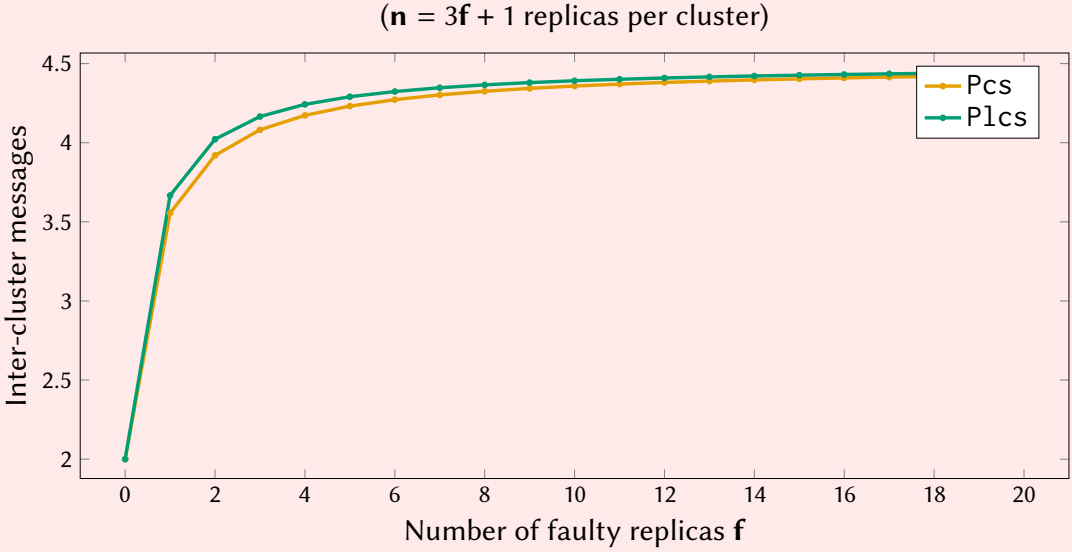
1: All replicas in $\mathrm{nf}(C_1)$ agree on v and construct $\langle v \rangle_{C_1}$.
2: **repeat**
3: Choose replicas $(\mathtt{R}_1, \mathtt{R}_2) \in C_1 \times C_2$, fully at random.
4: CS-STEP($\mathtt{R}_1$, $\mathtt{R}_2$, v).
5: *Wait* until the CS-STEP should finish.
6: **until** C_1 reaches consensus on $\langle \mathtt{proof}(\langle \mathtt{send}(v) \rangle_{C_2}) \rangle_{C_1}$.

Cluster-sending via probabilistic cluster-sending

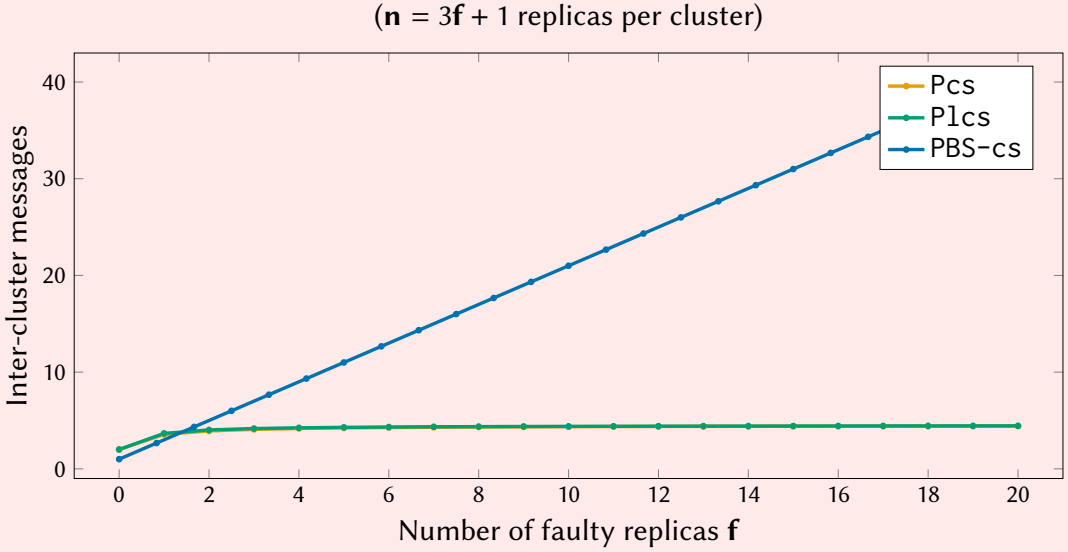
Protocol for the sending cluster C_1:

1: All replicas in $\mathrm{nf}(C_1)$ agree on v and construct $\langle v \rangle_{C_1}$.
2: Let $(S_1, S_2) := \Phi(C_1, C_2)$.
3: Choose $(P_1, P_2) \in \mathrm{perms}(S_1) \times \mathrm{perms}(S_2)$ fully at random.
4: $i := 0$.
5: **repeat**
6: cs-step$(P_1[i], P_2[i], v)$.
7: *Wait* until the cs-step should finish.
8: $i := i + 1$.
9: **until** C_1 reaches consensus on $\langle \mathtt{proof}(\langle \mathtt{send}(v) \rangle_{C_2}) \rangle_{C_1}$.

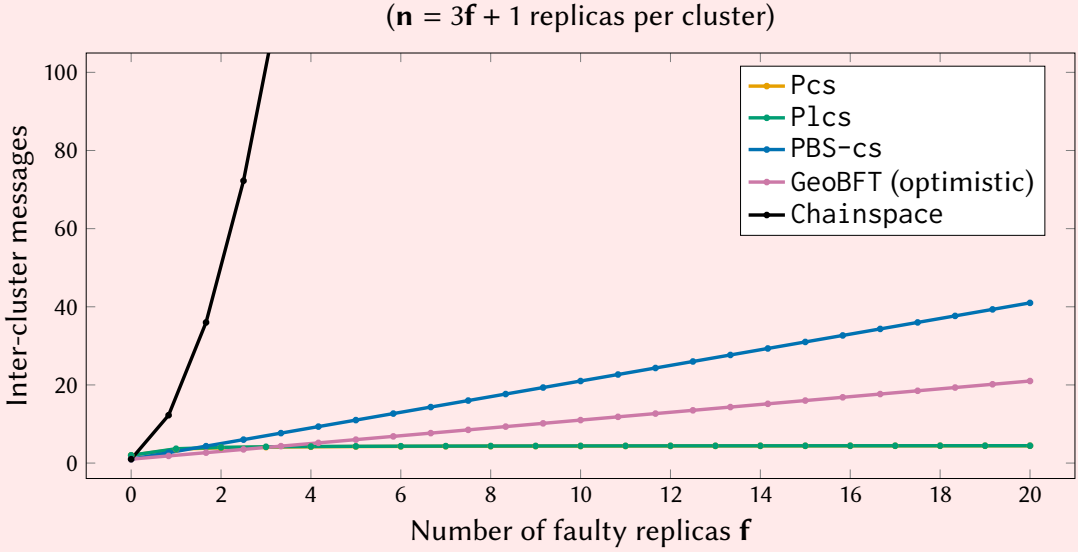
Performance of cluster-sending solutions



Performance of cluster-sending solutions



Performance of cluster-sending solutions



Cluster-sending in practice

Wide-area consensus GeoBFT uses *optimistic* cluster-sending techniques.

Cluster-sending in practice

Wide-area consensus GeoBFT uses *optimistic* cluster-sending techniques.

Sharded resilient databases ByShard, Cerberus, and RingBFT.

Cluster-sending in practice

Wide-area consensus	GeoBFT uses *optimistic* cluster-sending techniques.
Sharded resilient databases	ByShard, Cerberus, and RingBFT. AHL and Chainspace use cluster-sending *informally*.

Conclusion

More information

https://jhellings.nl

Technical reports:

- https://arxiv.org/abs/1908.01455.
- https://arxiv.org/abs/2108.08541.